þÿ One Line at a Time Generation and Internal Ev Interactive Poetry

Boggia, Michele

The Association for Computational Creativity
2022-06-27

Boggia , M , Ivanova , S , Linkola , S , Kantosalo , A & Toivonen , H 2022 , One Line at a þÿ Time Generation and Internal Evaluation of Interactive Poetry . in M Kantosalo , R Confalonieri , O Kutz & T Veale (eds) , Proceedings of the 13th International Conference on Computational Creativity . The Association for Computational Creativity , Bolzano , pp. 7-11 , International Conference on Computational Creativity , Bolzano , Italy , 27/06/2022 .

http://hdl.handle.net/10138/350144

unspecified
publishedVersion

One Line at a Time — Generation and Internal Evaluation of Interactive Poetry

Michele Boggia[♡] **Sardana Ivanova**[♠] **Simo Linkola**[♠] **Anna Kantosalo**[♠] **Hannu Toivonen**[♠]
[♡] Department of Digital Humanities [♠] Department of Computer Science
University of Helsinki, Finland
{first.last}@helsinki.fi

Abstract

We present methods that produce poetry one line at a time, in a manner that allows simple interaction in human-computer co-creative poetry writing. The methods are based on fine-tuning sequence-to-sequence neural models, in our case mBART. We also consider several internal evaluation measures by which an interactive system can assess and filter the lines it suggests to the user. These measures concern the coherence, tautology, and diversity of the candidate lines. We empirically validate two of them and apply three on the mBART-based poetry generation methods. The results suggest that fine-tuning a pre-trained sequence-to-sequence model is a feasible approach, and that the internal evaluation measures help select suitable models as well as suitable lines.

Introduction

We propose methods that use sequence-to-sequence neural models to generate poetry one line at a time. We use them to implement a simple interaction pattern where the system iteratively produces a set of line candidates from which the user picks one, with the aim of making poetry writing easy and entertaining for novices.[1]

We also consider four objective evaluation measures to assess candidate lines especially in interactive poetry generation. While we suggest these measures elsewhere (Boggia et al. 2022) we have not evaluated or applied them before. In this paper, we empirically validate them and show how to use the measures in practical applications.

Poetry generation is a popular research topic in computational creativity and numerous methods have been proposed in the literature (Gonçalo Oliveira 2017). Interactive poetry generation where the software acts as an ìntelligent or creative partner has also been addressed by several scholars (Kantosalo et al. 2014; Kantosalo, Toivanen, and Toivonen 2015; Ghazvininejad et al. 2017; Oliveira et al. 2017; 2019; Clark et al. 2018).

Our poetry generator produces poetry one line at a time, based on the previous lines of the poem or, in the case of the first line, based on keywords given by the user. This approach supports different user interaction patterns where suggestions for continuation are requested from the system. In this paper, we assume the following simple interaction pattern of Casual Poetry Creation (Boggia et al. 2022).

At any time, the system provides a handful of candidate lines from which the user chooses one. The system then iteratively generates candidates for the next line based on the user's previous selections. Candidates for the first line are generated from keywords inserted by the user. The system can be easily adapted to allow more complex interaction patterns, such as allowing the user to edit the system outputs.

The generation of candidate lines in such an interactive setting should satisfy three criteria: (1) each candidate should be related to the previous lines in the poem, or to the keywords in the case of the first line; (2) each candidate line should be poetic; and (3) the set of candidates for the nth line should be diverse. In the next section, we present poetry generation methods that address points 1 and 2; in the following section, we use the internal measures to address points 1 and 3.

Poetry Generation with mBART

To generate poems line by line we leverage mBART (Liu et al. 2020), a denoising autoencoder pre-trained on monolingual corpora in several languages. In a nutshell, the model takes a *source sequence* (e.g., a partially written poem) and produces a *target sequence* (the next line of the poem).

Starting from the same base model, we fine-tune (i) a model to generate candidates for the first poem line from the input keywords, and (ii) a model that generates candidates for additional poem lines. We will refer to these neural models as *first-line* and *next-line model* respectively.

The fine-tuning datasets for our models are constructed from the *Gutenberg Poetry Corpus*[2], a corpus of approximately 3M poem lines in English language extracted from Project Gutenberg[3].

First-Line Model

In our interaction pattern the first line of a poem is generated based on a small, unordered set of input keywords provided

[1] A Python implementation of the system is available at https://github.com/bmichele/poetry_generation.

[2] https://github.com/aparrish/gutenberg-poetry-corpus

[3] https://gutenberg.org

Proceedings of the 13th International Conference on Computational Creativity (ICCC'22)
ISBN: 978-989-54160-4-2

by the user.

In the fine-tuning step, we use the first lines of stanzas in the corpus as target texts. Since we do not have keywords for the poems or stanzas we obtain keyword proxies from the first lines by selecting two or more random content tokens among the nouns, adjectives and verbs on the line. The source text for each fine-tuning example is obtained by shuffling and concatenating the tokens.

Next-Line Models

At every iteration, after a line is selected by the user, the system should provide a set of candidates for the next line of the poem. Since there is no clear prescription on the best way to generate additional lines for a poem, we consider several options for fine-tuning mBART. We start by considering the previous line only, and progressively move towards more complex strategies. This allows us to compare candidate lines generated with different degrees of context. In general, we expect to obtain more surprising outcomes when the generation is based on a lower amount of textual input.

The first model we consider, *Next-Line Single*, is fine-tuned as follows: we iterate over all the lines in the corpus and build examples taking a poem line and its subsequent line as source and target sequence, respectively. We do not expect this model to produce lines that remain coherent with the user keywords after the first few iterations.

To get more coherent verses, we train two additional models: "Next-Line Multi" and "Next-Line Keywords". The *Next-Line Multi* approach fine-tunes mBART by using up to three consecutive poem lines as source sequence; the target is the verse following the input lines. The *Next-Line Keywords* approach increases coherence by conditioning next-line generation on the keywords obtained from the user.

The fine-tuning data is similar to the *Next-Line Single* model; for the Next-Line Keywords model we additionally prepend to the source sequence a pair of words related to the target sequence. To obtain them we first compute the average word vector of the target sequence tokens using Word2vec (Mikolov et al. 2013a; 2013b). We then retrieve the ten closest words in the Word2vec model by cosine similarity, and randomly sample two of them.

The fine-tuning strategies described above rely on the assumption that the base model, when fine-tuned over poem lines, will naturally learn to produce poetic line candidates. However, there is no control over how this is learned by the models and it will be influenced by the data that is present in the original corpus.

In the *Next-Line Rhyme* case we fine-tune a model that tries to generate, given a word and a line, a new poem line rhyming with the given word. Giving the word separately allows to produce lines rhyming with earlier lines, not just the previous one.

The fine-tuning data is similar to the data used for the *Next-Line Single* model, but we prepend to the source sequence a word rhyming with the target sequence; we use the CMU Pronouncing Dictionary[4] to look up rhymes. When no rhymes are found, we discard the pair. If multiple rhymes are available, we randomly sample up to four examples.

We fine-tune all the models for 10 epochs using batches of 64 examples over 4 GPUs. Due to the different preprocessing steps taken to build the fine-tuning data, the size of the datasets are slightly different for each model, resulting in a number of fine-tuning steps that is between 90k and 95k. We save model checkpoints every 15k steps.

Decoding Strategy

A good set of candidate lines is diverse, offering the user a real choice. An autoencoder such as mBART produces output sequences stochastically, so several candidates can be generated from the same model.

We generate candidates by sampling multiple sequences from the probabilities predicted by the fine-tuned models. In this way, the output sequences do not follow a distribution of high probability next tokens but are less predictable and can surprise the user (Holtzman et al. 2020). The randomness — and diversity — of the output can be controlled by the *temperature* parameter: values larger than one will increase the likelihood of low probability tokens.

Internal Evaluation Measures for Poetry

We consider four evaluation measures to assess the lines produced by the above poetry generation models. These measures can be utilised both (1) by the designer of the system during the system development to assess the feasibility of the generation methods and (2) by the system itself during its execution time to make informed decisions about which set of generated line candidates to show to the user.

Measures

We give a brief overview of the measures here. See our parallel paper (Boggia et al. 2022) for details.

We define the *n-Semantic Coherence* of a candidate for the ith line of the poem as follows. We consider the n previous lines, i.e., lines $i-n$ to $i-1$, transform them to a vector representation and compute its cosine similarity with a vector representation of the candidate line. Both vector representations are obtained computing the centroid of word vectors from the Word2Vec model. The idea is that the two vectors encode the semantic of the last lines of the poem and the candidate, respectively, and that their cosine similarity captures the degree of semantic similarity.

We define *Topic Coherence* of a candidate line as the cosine similarity between the vector representation of the line and the average of the word embeddings of the keywords used to generate the first poem line. The idea is to extend the concept of semantic coherence defined above and make it suitable to our interaction pattern in order to control the topic drift of each candidate from the initial keywords.

For a candidate line, we define *Tautology* as the number of tokens that are shared between the candidate and the previous line of the poem, normalized by the total number of tokens in the two lines. We consider this metric as our sequence-to-sequence models are based on mBART, which is pre-trained on denoising tasks and can be prone to copy

[4] https://github.com/cmusphinx/cmudict

the input sequence if not fine-tuned properly. Semantic coherence and tautology are likely to be correlated since both measure a similarity between consecutive lines. The former aims, however, at comparing meanings, while the latter compares words. An incoherent poem will display low semantic coherence scores; a high tautology value will indicate a repetitive poem.

We want our candidate poem lines on each generation step to be sufficiently different from each other, to give the user a true choice. We define the *diversity* of a set of lines as the average dissimilarity between the lines in the line set, where dissimilarity between two word sets is the additive inverse of the normalized word overlap between the sets.

Validation Datasets

We next validate the evaluation measures introduced above, showing that they do indeed measure what they are supposed to measure. Given the lack of suitable labelled data that could be used directly for this purpose, we resort to resampling lines from real poems. We can then compare the values of the measures on original poems against those obtained for random pseudo-poems.

To validate the measures, we use a dataset of poems from *Kaggle*, containing 6321 poems.[5] While the Poetry Corpus introduced in the previous section is suitable for training neural models, it is not optimal to validate the metrics of this section as it contains noisy data and there is no separation between poems.

Starting from the original poems, here referred to as *Real Poems*, we prepare two additional poem datasets. We randomly shuffle lines *within* each of the real poems and obtain the *Shuffled Poems* dataset. We then build the *Mixed Poems* dataset by shuffling lines *between* poems. To ensure equal poem line counts in the mixed poems we compute the poem lengths from each real poem and construct poems with the same number of lines by sampling (without replacement) from all the available poem verses in the corpus.

Validation of the Measures

The four metrics described above can be divided into two classes based on their implementation. First, semantic and topic coherence make use of word vectors to map textual inputs in a semantic space and compare them by computing the cosine similarity. Second, tautology and diversity are based on word overlap and rely solely on token sets. In this paper we validate the semantic coherence and diversity measures and argue that topic coherence and tautology will display a similar behaviour.

To validate the n-semantic coherence we consider the datasets with real, shuffled and mixed poems. Our hypothesis is that we should observe decreased semantic coherence scores after shuffling poem lines within a poem, and much lower values when considering pseudo-poems obtained by combining lines from random poems.

For each dataset we compute the n-semantic coherence of all poem lines (excluding first lines) with the n previous lines up to the first one or $n = 10$. Finally, we average the semantic coherence values for each order n (Figure 1). The average values of n-semantic coherence scores are systematically smaller when shuffling poem lines, with the lowest average values obtained when poems are composed of random lines.

[5] https://www.kaggle.com/michaelarman/poemsdataset

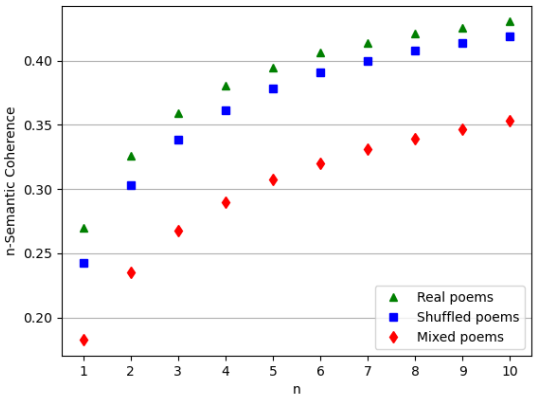

Figure 1: n-semantic coherence scores of Real Poems, Shuffled Poems and Mixed Poems as a function of n, the number of previous lines considered.

Temperature	Diversity	Diversity without stopwords
1	0.1406	0.1039
2	0.4014	0.3429
3	0.4788	0.4232
4	0.5174	0.4434
5	0.5377	0.4701

Table 1: Average diversity scores of sets of candidate lines as a function of temperature, a generation-time parameter affecting diversity.

To inspect how the diversity measure behaves when computed over different sets of poem line candidates, we rely on synthetic data produced by our first-line model. We construct 100 random keyword pairs by combining nouns and verbs sampled from a list of common English words. For each keyword pair, we use different temperature parameter of the model to generate multiple batches of ten candidates each. Candidates generated with higher temperatures are more diverse by construction.

As expected, the diversity increases as a function of the temperature (Table 1). This is true for the full lines (middle column) as well as when stopwords have been removed before computation of diversity (right column). This validates that the diversity measure does catch differences in the generated lines.

Analysis of the mBART-Based Methods with Internal Evaluation Measures

We now apply the internal evaluation measures on the mBART-based poetry generation methods. With this brief

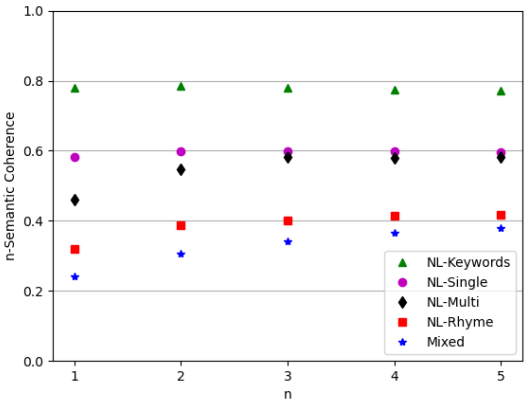

Figure 2: n-semantic coherence scores for poems generated by different Next-Line models, as a function of n, the number of previous lines considered.

Model	Tautology ↓	Diversity ↑
NL-Keywords	0.206	0.184
NL-Single	0.145	0.431
NL-Multi	0.096	0.459
NL-Rhyme	0.046	0.352
Mixed	0.045	0.841

Table 2: Average tautology and diversity scores for line candidates generated using different Next-Line models. For tautology, smaller values are better; for diversity, larger.

example we aim to shed light on the specific generation methods, as well as to illustrate how the evaluation measures can be used to assess generation methods. In this experiment, we compare the four flavours of the Next-Line model previously described, using the evaluation measures for semantic coherence, tautology and diversity.

In order to test the generation methods without user interaction, we generate poems automatically by sampling the first poem line from a dataset of real poems and then selecting a random candidate line at each generation step. We stop the generation process after ten candidate selections. To collect data for diversity assessment, we also log the line candidates proposed by the generator at each iteration. We use the above procedure to obtain 100 poems and 1000 sets of candidates with each of the four Next-Line models.

As a baseline, we fine-tune a model over random poem lines from the Gutenberg Poetry Corpus both as source and target sequences. This model, called *Mixed*, gives a lower bound for the coherence of poems.

We report the n-semantic coherence scores of the resulting poems in Figure 2, and their tautology and diversity scores in Table 2. Based on these results we can make several observations about the models, such as the next two.

The *NL-Keywords* model, introduced to avoid topic drift, effectively improves the coherence of the poems (Figure 2), but the price to pay is that poems become repetitive and have low diversity (Table 2). A qualitative inspection of the generated poems confirms this finding. For instance, this poem was obtained with keywords "bury" and "dead":

> *And let the dead folk bury their dead,*
> *But let the dead men bury their dead,*
> *Let the dead men bury their dead,*
> *Let the living men bury their living,*
> *Let the dead folk sleep.*
> ....

The *NL-Multi* model, on the other hand, produced relatively interesting poems even without human interaction:

> *Which tries, and counter-stands the shock,*
> *Of time and chance;*
> *And, having learn'd to bear the yoke,*
> *To bear the yoke must learn to love,*
> *And follow Truth, and all that's above.*
> *The ways that lead to Heaven's high throne,*
> *Are long and hard to tell;*
> *But this way leads to God alone,*
> *And that way leads to Hell.*

The success of the ML-Multi model in this respect is no surprise: it obtained both high semantic coherence scores as well as a high diversity score.

A different but important application for the internal measures is the optimization of the set of candidates towards a desired feature. For instance, assume that the system fails to satisfy the user because of a lack of diversity in the candidates. The sequence-to-sequence models could then be used to generate a larger number of potential candidates (which in our setup is computationally inexpensive), and they could then be narrowed down to a final set of candidates while maximising their diversity.

Conclusion

We gave several variants of fine-tuned mBART-models for line-by-line poetry generation. One variant produces opening lines from keywords, while other models produce lines to continue a partially written poem. The models consider varying contextual information: one or more previous lines, user-given keywords, or rhyme. The methods are designed in a manner that should allow relatively easy adaptations to different genres, corpora, and even languages.

We empirically validated internal evaluation measures of lines of poetry. We showed that the proposed measures of coherence and diversity correlate with ground truth.

Finally, we applied three evaluation measures on generation methods that continue an incomplete poem. The results indicate trade-offs between the methods. The NL-Multi method that uses several lines as a context seems to strike a good balance.

The choice to work line-by-line, both in generation and in internal evaluation of poetry, stems from the desire to support Casual Poetry Creation (Boggia et al. 2022) and to make co-creative poetry writing as easy as possible. The next step is an evaluation of the approach with actual users.

Author Contributions

MB, SI and HT conceived the poetry generation methods. MB, SI, SL and HT selected and defined the internal evaluation measures. MB and SI processed the raw data for the experiments. SI fine-tuned the first-line model, MB the next-line models. MB and SL implemented and validated the metrics. MB was in charge of writing the technical part; everybody contributed to the writing of the other sections.

Acknowledgments

MB has been partially supported by the ERC Consolidator Grant FoTran (agreement № 771113) and by the grant 825153 (EMBEDDIA) under the European Union's Horizon 2020 research and innovation programme.
SI has been funded by the European Union's Horizon 2020 research and innovation program under grants 825153 (EMBEDDIA) and 770299 (NewsEye). SL and AK have been funded by Academy of Finland (Grant #328729).

References

Boggia, M.; Ivanova, S.; Linkola, S.; Toivonen, H.; and Kantosalo, A. 2022. Casual Poetry Creators: A design pattern and internal evaluation measures. In *Proceedings of the 13th International Conference on Computational Creativity*. ACC.

Clark, E.; Ross, A. S.; Tan, C.; Ji, Y.; and Smith, N. A. 2018. Creative writing with a machine in the loop: Case studies on slogans and stories. In *Proceedings of the 23rd International Conference on Intelligent User Interfaces*, IUI '18, 329–340. New York, NY, USA: ACM.

Ghazvininejad, M.; Shi, X.; Priyadarshi, J.; and Knight, K. 2017. Hafez: an interactive poetry generation system. In *Proceedings of The 55th Annual Meeting of the Association for Computational Linguistics, System Demonstrations*, 43–48. ACL.

Gonçalo Oliveira, H. 2017. A survey on intelligent poetry generation: Languages, features, techniques, reutilisation and evaluation. In *Proceedings of the 10th International Conference on Natural Language Generation*, 11–20. Santiago de Compostela, Spain: ACL.

Holtzman, A.; Buys, J.; Du, L.; Forbes, M.; and Choi, Y. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations (Poster Presentation)*.

Kantosalo, A.; Toivanen, J. M.; Xiao, P.; and Toivonen, H. 2014. From isolation to involvement: Adapting machine creativity software to support human-computer co-creation. In *Proceedings of the Fifth International Conference on Computational Creativity*, 1–7. Jožef Stefan Institute.

Kantosalo, A.; Toivanen, J. M.; and Toivonen, H. 2015. Interaction evaluation for human-computer co-creativity: A case study. In *Proceedings of the Sixth International Conference on Computational Creativity*, 276–283. Brigham Young University.

Liu, Y.; Gu, J.; Goyal, N.; Li, X.; Edunov, S.; Ghazvininejad, M.; Lewis, M.; and Zettlemoyer, L. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics* 8:726–742.

Mikolov, T.; Chen, K.; Corrado, G. S.; and Dean, J. 2013a. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations (Workshop Presentation)*.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*. NIPS.

Oliveira, H. G.; Hervás, R.; Díaz, A.; and Gervás, P. 2017. Multilingual extension and evaluation of a poetry generator. *Natural Language Engineering* 23(6):929–967.

Oliveira, H. G.; Mendes, T.; Boavida, A.; Nakamura, A.; and Ackerman, M. 2019. Co-PoeTryMe: interactive poetry generation. *Cognitive Systems Research* 54:199–216.